Qualcomm

Seattle, WA

16 - 18 Nov 2024

Deferred Memory Block Init for Boot Time Reduction

Sudarshan Rajagopalan

Engineer, Staff
Qualcomm Innovation Center, Inc.

quic_sudaraja@quicinc.com

Agenda

1. About me and the team
2. paging_init, bootmem_init and mm_init
3. Deferred Memory Initialization
4. Limiting Boot Memory
5. Boot Time Reduction
6. Challenges and Limitations

About me

- Linux Kernel Developer in the Linux Kernel team at Qualcomm Innovation Center, Inc. in San Diego, California.
- Interests are in Linux Kernel Memory Management, Embedded Systems and Firmware, System designs and Computer Architecture.
- PS: we are hiring to expand the team!

paging_init

paging_init

- **paging_init** establishes the foundation of memory management in the Linux kernel.
- Called as part of **setup_arch** during early boot process.
- Sets up the page-tables for all the System RAM memory.
- All memory initialization happens in **single thread** in single **boot CPU**.
- On machines with very large RAM (few TBs), paging_init could take considerable amount of time thereby increasing bootup time.

bootmem_init
mm_init

bootmem_init

mm_init

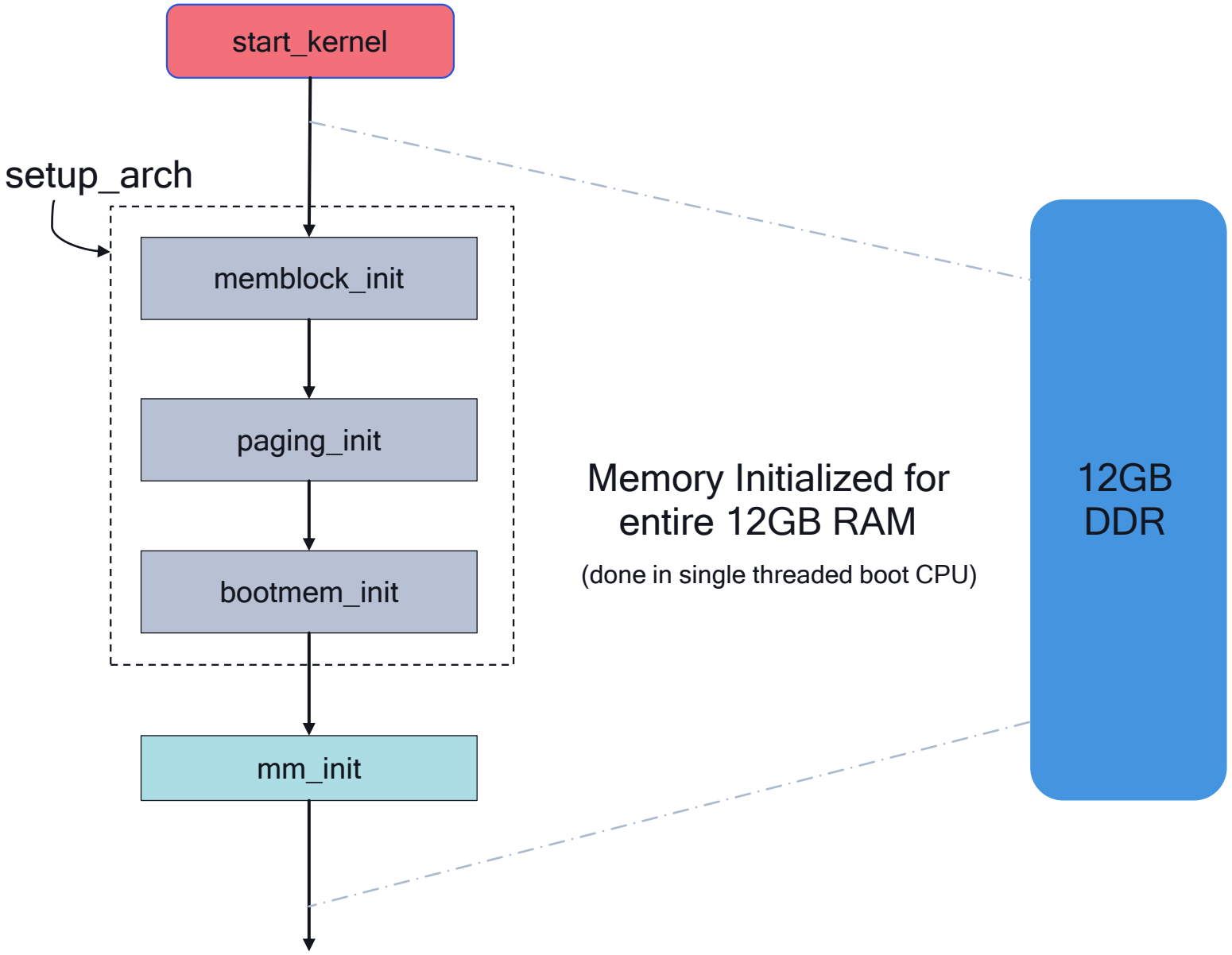
bootmem_init

- Initializes all the memory blocks by creating the **memmap meta data** (aka struct page memory).
- Responsible for **sparse mem init** – initializes all sections in memory node and populates memmap data for all sections
- Part of **setup_arch** as well.

mm_init

- Initializes all the kernel memory allocators –
  - `stack_depot_early_init()`, **`mem_init()`**, `kmem_cache_init()`, `kmemleak_init()`, `vmalloc_init()`, `mm_cache_init()`.
  - **`mem_init()`** adds all memory into page allocator freelist using `memblock_free_all()`. Execution time depends on System RAM size.

System Memory Initialization



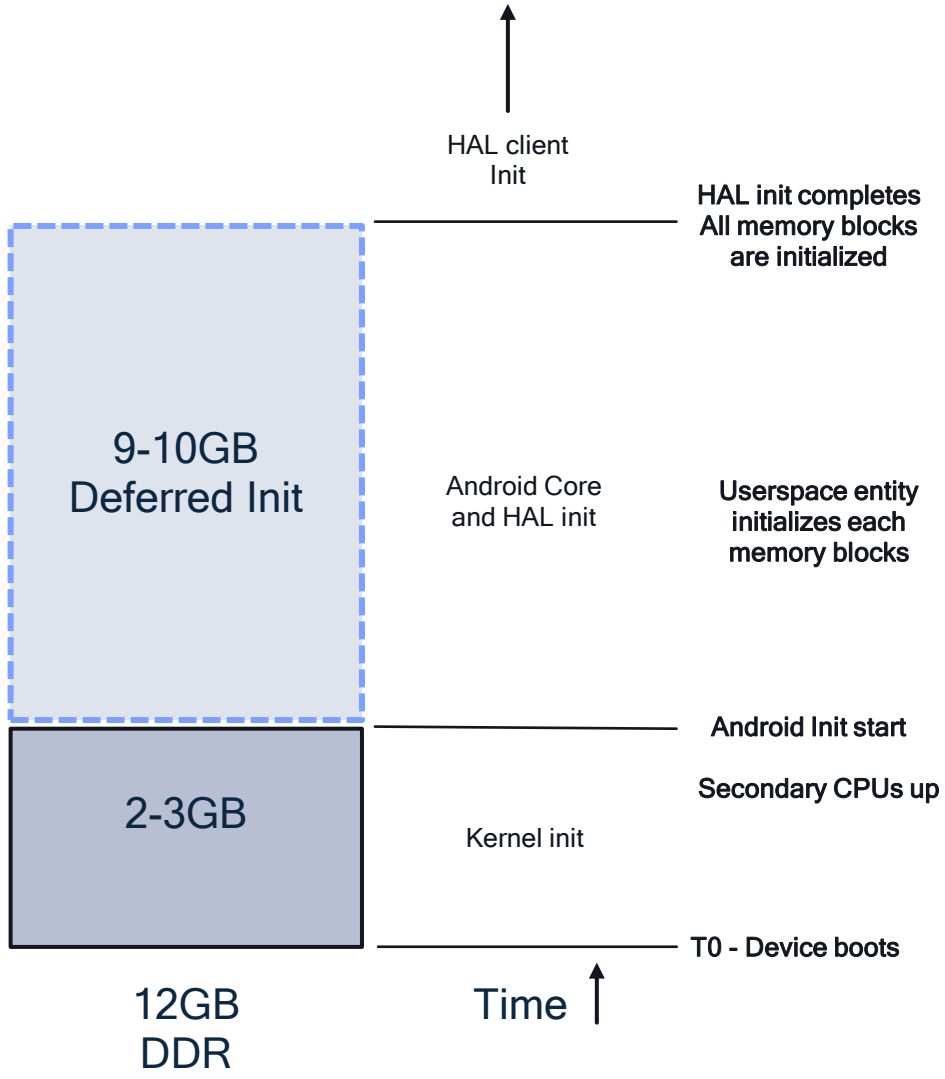
Deferred Memory Initialization

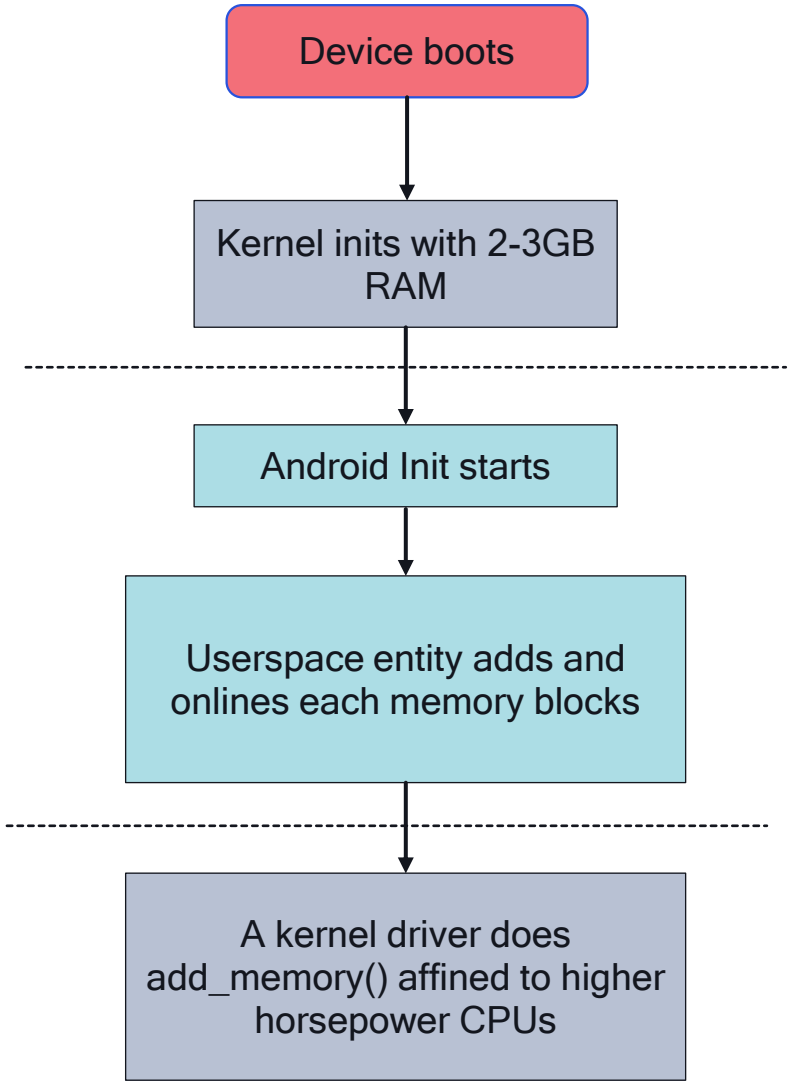
Deferred Memory Initialization

Motivation

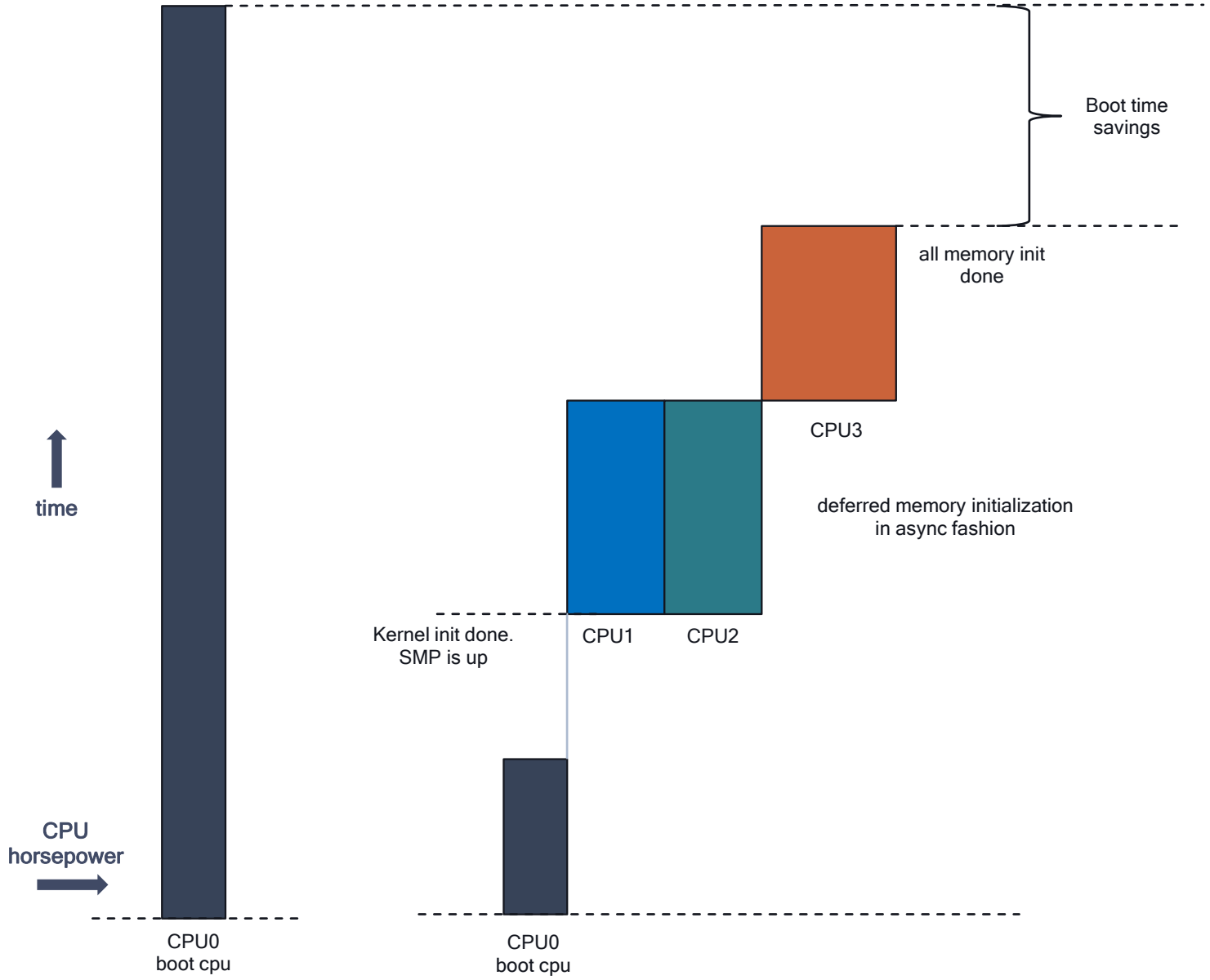
- In systems with huge RAM to reduce overall boot time KPI, we can bring up a subset of memmap during boot sufficient for kernel to finish initialization
- Once kernel init is done and SMP is up, then initialize rest of the memory in parallel fashion.
- Originally inspired by config DEFERRED_STRUCT_PAGE_INIT for servers.
- Initialization of remaining memory for different zones in different memory nodes can be done with different kthreads affined to bigger CPUs for faster memory init.
- Might impact potential performance of tasks running early in the lifetime of the system until these kthreads finish the initialization.
- This overall saves several milliseconds of boot time KPI per GB of memory initialized later after kernel init.

Deferred memory init for boot time optimization





Deferred Memory Initialization



Adding memory to required zones

- Some systems might have different memory zone configuration – ZONE_NORMAL, ZONE_DMA, ZONE_MOVABLE etc.
- Example, a system with 12GB RAM would have the below configuration –
  - 2GB DMA zone, 5GB Normal zone, 5GB Movable zone, single NUMA node
- Kernel kthreads would have to know about these configurations.
- Instead, once kernel init is done, a userspace process can add the remaining blocks into required memory Zone by using the memory hotplug framework.
- Memory added after boot aren't considered as "boot memory" but rather driver managed memory.
- Flexibility to add memory into required zone –
  - `echo addr > /sys/devices/system/memory/probe`
  - `echo online_movable > /sys/devices/system/memory/memoryXXX/state`
  - `echo online_kernel > /sys/devices/system/memory/memoryXXX/state`

Limiting Boot Memory

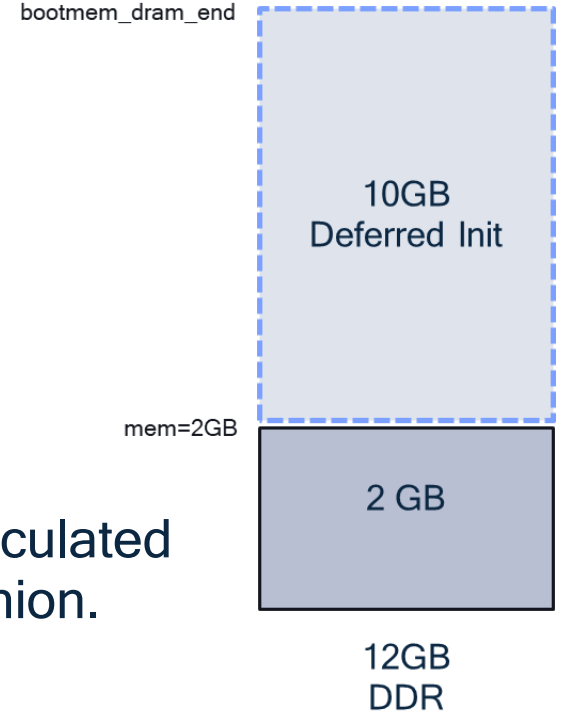
Limiting boot memory

‘mem=‘ cmdline parameter

- Limit the kernel boot memory using ‘mem=‘ cmdline parameter
- Calculate dram size using “memory” node in DT.

```
/ {
   #address-cells = <2>;
   #size-cells = <2>;
   memory { device_type = "memory"; reg = <0x0 0x0 0x3 0x00000000>; };
```

- Remaining memory = dram_size – bootmem
- Kernel will bootup with mem=XGB bootmem
- Rest of the remaining memory can be calculated as above and initialized later in async fashion.
- Gives flexibility to add the remaining memory into required memory zones and nodes.



Ways of Deferred Memory Initialization

1.

Using kernel driver

On late_initcall

- During late_initcall, a kernel driver could get the **mem=** and dram_end_addr information and use **memory hotplug** to add the remaining memory blocks
- `add_memory_driver_managed()` – adds special, driver-managed memory to the system as system RAM.
- Memory hotplug cmdline parameters –
  - `memhp_default_state=online` – auto-onlining when memory is added.
  - `movable_node` – when set, the kernel will default to Movable zone when onlining a block.
- Limitation of using kernel driver – add_memory can only add to specific memory zone and cannot be changed runtime.

2.

Using userspace service

Post kernel init

- A kernel driver could get the bootmem_end_addr and dram_end_addr information and expose these info to userspace via sysfs nodes.
- DT property that mentions amount of Normal/Movable zone memory to be added in deferred fashion.

```
deferred_mem_init {
        movablecore-size = < 0x1 0x40000000>;
};
```

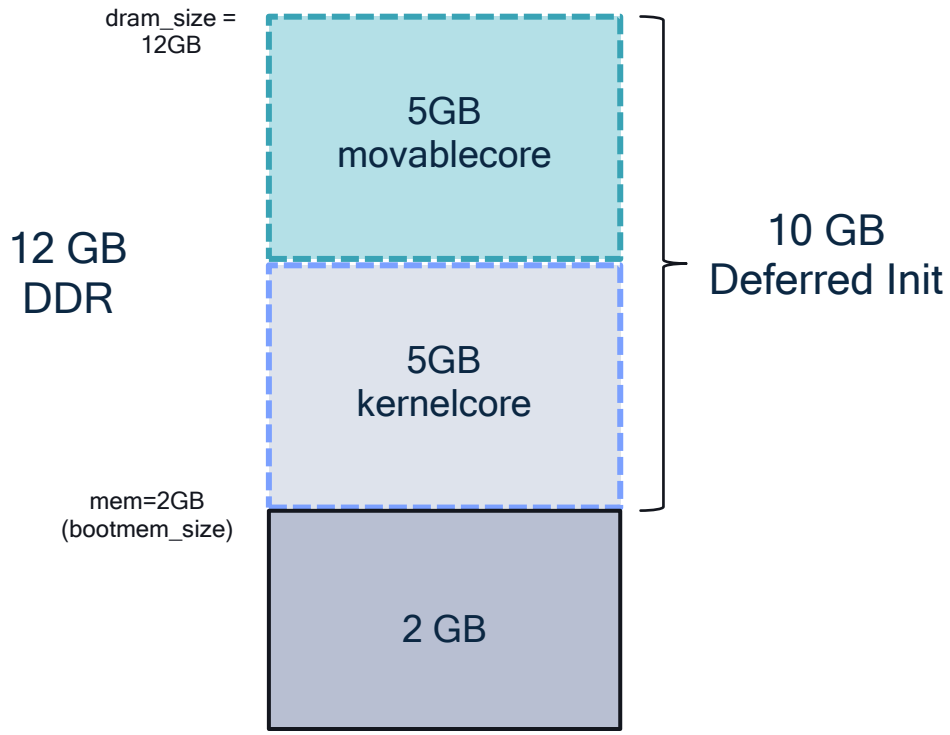
  - `sys/kernel/deferred_mem_init`
  - `sys/kernel/deferred_mem_init/memblock_end_of_dram`
  - `sys/kernel/deferred_mem_init/bootmem_end_of_dram`
  - `sys/kernel/deferred_mem_init/deferred_kernelcore`
  - `sys/kernel/deferred_mem_init/deferred_movablecore`
- Gives flexibility to add memory into required memory Zone.

2.

Using userspace service

Post kernel init

- `bootmem_size` = 2GB (mem=2GB)
- `dram_size` = 12GB (12GB total System RAM)
- `movablecore-size` = 5GB (from DT property)
- Remaining memory = 12 – (2 – 5) = 5GB = Kernel memory (kernelcore-size)



Boot time Reduction

Boot KPI Reduction

- On Qualcomm® SM8550 with 12 GB DDR, booting with 2GB bootmem we have seen about **~160-200ms** boot KPI savings by adding 8GB in deferred async fashion.
- **~20-30ms** reduction in boot KPI per GB.
- Observed by profiling paging_init and bootmem_init.
  - Note that this time is invisible for kernel timestamp, since pagint_ing, bootmem_init, and mm_init is before time_init.
  - So we have to use other hardware timestamps/clock for measuring overall kernel bootup time.

Challenges and Limitations

Challenges and Limitations

- Requires **memory hotplug** support to be enabled.
- Full parallelism cannot be attained due to **zone_lock** being held per zone while doing add_memory.
- Determining initial bootmem size sufficient to bootup kernel or userspace.
- Capturing boot time savings with time measurements is hard since kernel time isn't initialized before paging_init.
- Robust code and error handling – make sure that system is up with entire dram size even in presence of errors in deferred memory init mechanism.
- Might interfere with performance of tasks running early in the lifetime of the system.

Upstream pathway

Upstream pathway

- Feature isn’t upstreamed yet. Planning to send patches as RFC to the upstream community.
- Kernel method of deferred memory initialization -- so as it doesn’t rely on any user space implementation.
- Fix the limitation in Kernel method by allowing add_memory to required zone.
  - `add_memory_zone()` ?
  - Has its own challenges and limitations – making sure memory zones are not discontinuous.
- Implement as separate kernel driver

  mem=XGB deferred_mem.kernelcore=nn% deferred_mem.movablecore=nn%

  - Example for 12GB DDR: `mem=2GB, deferred_mem.movablecore=40%` -- 2GB bootmem (Normal zone), 5GB deferred added to Normal zone, 5GB deferred added to Movable zone.

Questions

Thank you

Qualcomm

Follow us on:

For more information, visit us at:

qualcomm.com & qualcomm.com/blog